

## 활 동 보 고 서

회차	7회차	지도교수 확인란	
일시	2019.9.18.(수) 19:00~22:00		
장소	충남대 경상대 스터디룸 1층		
주제	JAVA(3) 클래스 인스턴스 그리고 객체, 클래스 멤버와 인스턴스 멤버~패키지까지		
학습 내용 요약			
<p>&lt;클래스와 인스턴스 그리고 객체&gt;</p> <p>1) 메소드화</p> <p>프로그래밍의 기본은 중복을 제거하는 것이다. 중복을 제거하기 위한 방법 중 하나가 메소드이다.</p> <p>2) 클래스</p> <p>: 클래스는 연관되어 있는 변수와 메소드의 집합이다.</p> <p>‘생활코딩’에서는</p> <pre>class Calculator {</pre> <p>라는 예시를 들었다.</p> <p>로직들을 대표하는 이름을 계산기라는 의미의 Calculator라고 정하고 이것들을 Calculator이라는 이름으로 그룹핑한 것인데, 이럴 때 사용하는 키워드가 class이다.</p> <p>class 키워드 뒤에는 클래스 이름이 오고, 그 뒤에는 중괄호가 온다.</p> <p>중괄호는 클래스의 시작과 끝의 경계를 의미한다.</p> <p>3) 인스턴스</p> <p>일종의 설계도인 ‘클래스’를 정의하는 것으로는 할 수 있는 것이 많지 않기 때문에, 그 때 사용하는 키워드가 new이다.</p> <pre>Calculator c1 = new Calculator();</pre> <p>new Calculator()은 클래스 Calculator를 구체적인 것으로 만드는 명령이다. 이렇게 만들어진 구체적인 것을 인스턴스(instance)라고 부른다.</p> <p>&lt;클래스 멤버와 인스턴스 멤버&gt;</p> <p>1) 멤버</p> <p>멤버member는 영어로 구성원이라는 뜻이며, 객체 또한 이러한 구성원을 가지고 있다.</p>			

객체를 만들기 위해서는 우선 클래스를 정의하고, 클래스에 대한 인스턴스를 만드는데, 특정 파일에서 각 변수(예를 들면 left, right 변수)는 인스턴스의 멤버다.  
인스턴스는 각자 다른 값을 가지고 있다.



## 2) 클래스 변수

예시를 들었던 left의 값은 인스턴스마다 달라질 수 있다. 인스턴스의 상태인 변수의 값이 인스턴스마다 다른 값을 가질 수 있다는 점은 하나의 클래스를 여러 개의 인스턴스로 만들어서 사용할 수 있다는 것을 의미한다. 또한 모든 인스턴스가 같은 값을 공유하게 하고 싶다면 각각의 변수를 클래스의 멤버로 만들면 된다.

## 3) 클래스 메소드

합계나 평균을 구할 때마다 좌향과 우향의 값을 주는 방식으로 계산을 할 수도 있다.

## 4) 용어

인스턴스 변수와 클래스 변수는 아래와 같이 부르기도 한다.

\*인스턴스 변수 -> Non-Static Field

\*클래스 변수 -> Static Field

## <유효범위>

변수와 메소드 같은 것들을 사용할 수 있는 것은 이름이 있기 때문이다. 아래 코드에서 left는 변수의 이름이고, sum은 메소드의 이름이다.

```
1 int left;  
2 public void sum() {}
```

(이해를 돕기 위한 '생활코딩' 사이트 영상 발췌)

프로그램이 커지면 여러 가지 이유로 이름이 충돌하게 된다. 이를 해결하기 위해서 고안된 것이 유효범위라는 개념이다. 흔히 스코프(Scope)라고도 부른다.

### <초기화와 생성자>

#### 1) 초기화

어떤 일을 시작하기 전에 준비를 하는 것 = 초기화

#### 2) 생성자

- 값을 반환하지 않는다.

생성자는 인스턴스를 생성해주는 역할을 하는 특수한 메소드라고 할 수 있다. 그런데 반환 값이 있다면 엉뚱한 객체가 생성될 것이다. 따라서 반환 값을 필요로 하는 작업에서는 생성자를 사용하지 않는다. 반환 값이 없기 때문에 return도 사용하지 않고, 반환 값을 메소드 정의에 포함시키지도 않는다.

- 생성자의 이름은 클래스의 이름과 동일하다.

자바에서 클래스의 이름과 동일한 메소드는 생성자로 사용하기로 약속되어 있다.

### <상속>

상속(Inheritance)이란 물려준다는 의미다. 어떤 객체가 있을 때 그 객체의 필드(변수)와 메소드를 다른 객체가 물려받을 수 있는 기능을 상속이라고 한다.

### <overriding>

-창의적인 상속

상속은 상위 클래스의 기능을 하위 클래스에게 물려주는 기능이다. 그렇다면 하위 클래스는 상위 클래스의 메소드를 주어진 그대로 사용해야 할까? 만약 그래야 한다면 제약이 상당할 것이다. 이런 제약을 벗어나려면 하위 클래스가 부모 클래스의 기본적인 동작방법을 변경할 수 있어야 한다. 이런 맥락에서 도입된 기능이 메소드 오버라이딩(overriding)이다.

### <overloading>

메소드 오버로딩은 매개변수를 사용한다. 즉 매개변수가 다르면 이름이 같아도 서로 다른 메소드가 되는 것이다. 반면에 매개변수는 같지만 리턴타입이 다르면 오류가 발생한다.

### <패키지>

패키지(Package)는 하나의 클래스 안에서 같은 이름의 클래스들을 사용하기 위한 방법이다.

클래스가 많아짐에 따라서 같은 이름을 가진 클래스가 생겨날 가능성이 높아지게 되는데 이름의 충돌을 방지하기 위한 고안된 것이 패키지라고 할 수 있다.

## 학습 방법 및 과정

7주차에는 6주차보다도 학습할 내용이 매우 많았다. 계획을 짤 때 목차만 보고 비슷한 분량인 줄 알았는데 내용까지 세부적이게 보고 더 철저하게 계획을 작성 할 걸 하는 후회가 약간은 들었지만 매도 먼저 맞는게 낫다 싶어서 계획표대로 10개의 파트을 다 하기로 했다. 그래서 학습 내용요약은 저번 보다 꽤 길지만 6주차처럼 서로 교류하는 시간은 조금은 부족하지 않았나 해서 싶었다.

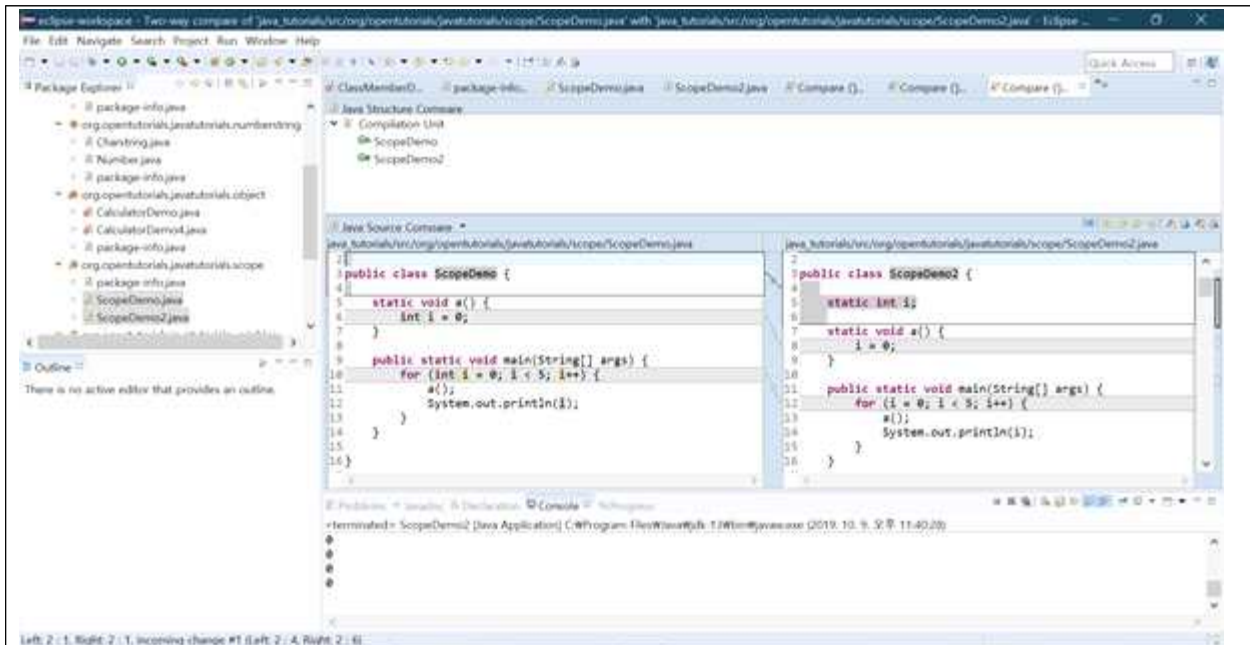
그리고 우리는 10개의 강의를 3시간 안에 다 드는 것이 버거웠기에(강의만 듣는 것이 아니라 서로 노트필기나 워드 정리도 하면서 들었고 자바 프로그램을 실행하면서 들어야 했기에 시간이 매우 역부족이었다.) 각자 자신의 페이스대로 분량을 조절해서 최저 3개~최고 6개의 강의를 들었다.

약속된 3시간 동안만 코딩을 하고, 할 일이 많은 학우는 숙제로 7주차의 분량을 공부해오기로 하고, 남아서 학습을 마치고 갈 수 있는 학우는 경상대 스터디룸에 남아 공부를 마무리 했다.

#### <학습소감>

[신예슬 학우]: 하나의 클래스를 가지고 여러가지 인스턴스, 즉 변수가 각기 다른 인스턴스들을 만들어서 재활용을 할 수 있다는 것이 객체지향프로그래밍을 이해하는데 도움이 되었다. 지난주에는 가장 기초적인 문법을 배웠다면, 이번 시간에는 여러 메소드를 오버로딩하고 오버라이딩하면서 활용하는 방법과 상속 등으로 여러가지 활용 가능성에 대해 알 수 있었다. 자바의 가장 중요한 문법인 클래스와 인스턴스를 배우면서 살짝 헷갈리기도 했지만 가장 중요한 만큼 몇번이고 돌려보느라 시간이 아주 많이 소요됐다. 강의에서 이해가 안가고 부족한 점은 도서관에 가서 자바 책을 몇 권 빌려서 읽고 또 읽었다. 예제도 따라해보고 실행해본 결과 어느정도 감을 익힌 것 같다. 또한, 유효범위가 왜 생겨났고, 어떨때 필요하며, 어떻게 작동하는지에 대해 배웠다. 유효범위 또한 객체지향프로그래밍을 이해하는데 중요한 부분인 것 같다. 유효범위에 대해 배우면서 객체지향프로그래밍이 어떻게 동작하는지, 왜 객체지향프로그래밍을 고안한 것인지 생각해볼 수 있었다. 유효범위가 없는 상황과 절차지향프로그래밍을 생각해 보면 객체지향프로그래밍에 대해 더 이해할 수 있게 되는데, 객체지향프로그래밍은 클래스를 통해 재활용성을 높이고, 정리할 수 있고, 유효범위라는 것이 있기 때문에 프로그래밍의 규모가 커져도 변수의 충돌을 막을 수 있으며 유지보수가 편리하고 좀 더 체계적으로 프로그래밍을 관리할 수 있게 된다. 그렇기에 아주 효율성이 높은 언어라고 볼 수 있다. 왜 다들 자바를 배우라고 하는지 조금 알 것 같았다. 파이썬과는 다르게 아주 까다롭고 번거롭지만 그만큼 처음에 공을 들여놓으면 나중에 유지 보수 하기가 편하고 더 복잡하고 큰 프로그램을 만들기에 아주 적합한 언어라고 생각한다.

[강미규 학우]: 유효범위를 배울 때, 사실 Demo1을 할 때와 Demo2를 할 때 뭐가 달라진 건지 잘 모르겠는데 실행하면 결과는 매우 다르게 나와서 혼란스러워 하고 있었다. 그런데 Compare with-Each other 기능을 실행하니 어떤 로직이 달라진 건지 한눈에 알 수 있어 편리하고 신기했다.





[이정란 학우]: 클래스를 타입별로 분류하고 그 차이를 구분하는 것이 은근히 어려웠다. 클래스 멤버, 클래스 메소드, 인스턴스 메소드가 있는데 인스턴스 메소드는 클래스 멤버에 접근할 수 없고, 클래스 메소드는 인스턴스 메소드에 접근할 수 없다고 한다. 이 이유를 파악하는 부분이 엄청 어려웠다.

[김예린 학우]: 클래스와 메소드, 인스턴스는 파이썬을 공부할 때 배운 부분이었다. 그래서인지 이해하기가 파이썬을 공부할 때보다는 조금 수월했으나 그래도 객체지향의 개념이 잘 와닿지 않았다. 또한 '생활코딩'의 자바 강좌는 파이썬과 루비 강좌와 다르게 코드를 작성할 시간을 주지 않고 기존의 코드를 불러와서 실습을 진행하는 경우가 많았기에 실습에 시간이 많이 소요되었다. 영상에 나온 코드를 타이핑하는 과정에서 오타가 나서 오류가 발생하는 경우도 있어서, 실질적으로 매소드나 인스턴스, 상속, 패키지 등의 실습에서 시간이 오래 걸렸다고보다는 코드를 정말로 '작성'하는 것 자체에 시간이 많이 소요되어 버거웠다.

[김현지 학우]: 변수를 선언할 때 종류에 따라 다르게 해야 한다는 점이 매우 불편했다. 루비파이썬에서 객체 지향 프로그래밍을 이미 한 번 접해봤기 때문에 이번에는 좀 더 나을 것 같았는데 아직도 와닿는 것이 없다. 코드를 입력하는 것도 더 번거롭고 길고 복잡하다. 기초 개념만 곁핍기 한다는 느낌이 강하고 자바로 뭘 할 수 있을 지 모르겠는데 공부하는 데 시간이 많이 걸렸다.

#### 활동 사진 첨부

<p>학습 활동</p>		<p>학습 결과</p>	<p>간단한 로직 (신예슬 학우의 이클립스 화면)</p>
--------------	--	--------------	---------------------------------

<p>①</p>		<p>물 ①</p>	<pre>public class CalculatorDemo {     public static void main(String[] args) {         // 아래 로직이 1000줄에 달하는 복잡한 로직이라 가정해보자.         System.out.println(10 + 20);         System.out.println(20 + 40);     } }</pre>
<p>학 습 활 동 ②</p>		<p>학 습 결 과 물 ②</p>	<pre>public class CalculatorDamo2 {     public static void sum(int left, int right) {         System.out.println(left + right);     }     public static void main(String[] args) {         sum(10, 20);         sum(20, 40);     } }</pre> <p>(역시 신예슬 학우의 이클립스 화면. 상단의 코드에서 +라는 중복을 제거한 로직은 같음.</p>